Factor Behavioral Alternatives

1. Overview

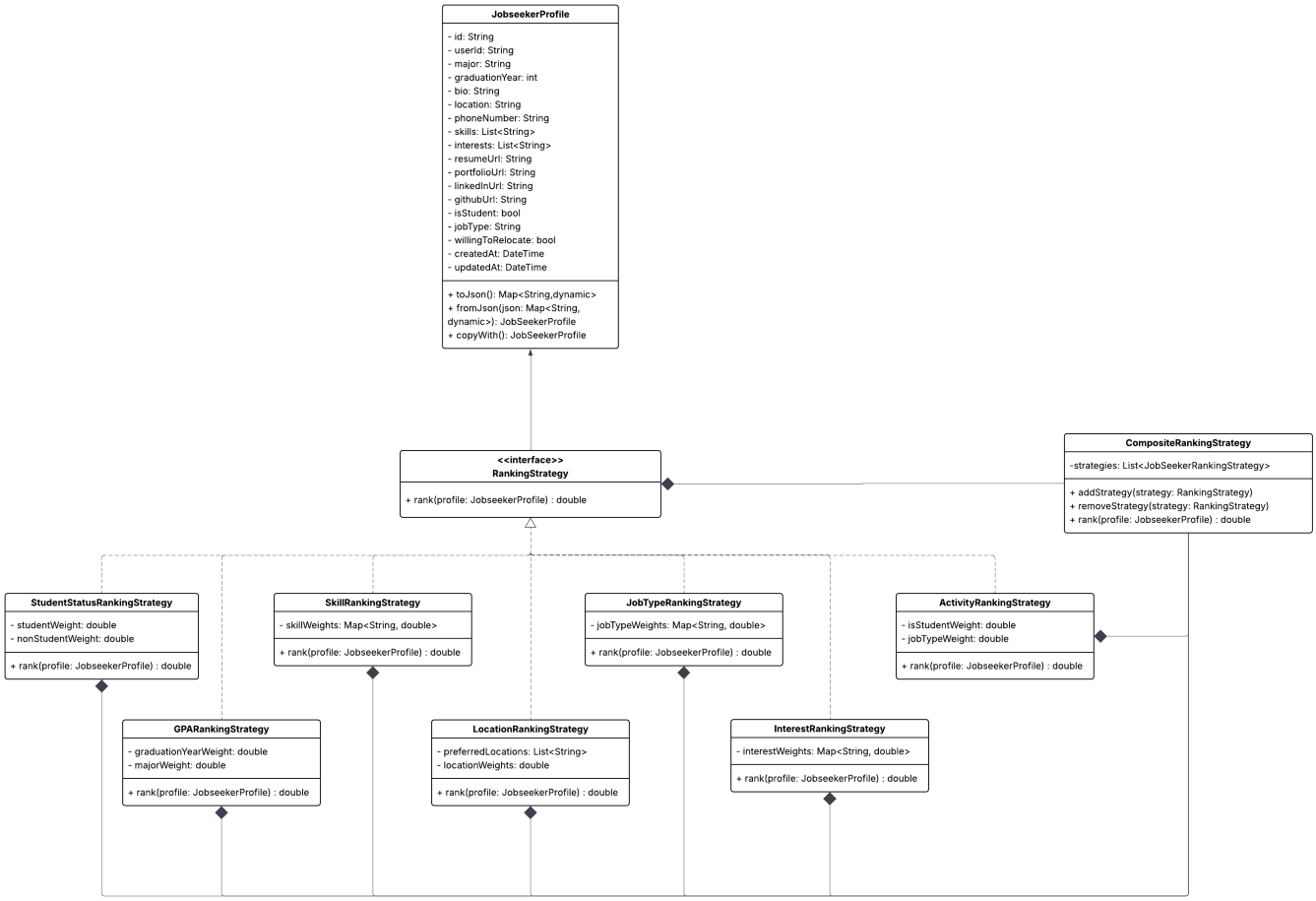
This document provides the UML rationale for the **Factor Behavioral Alternatives**, which evaluates profiles like Job Seekers, Students, among others, using multiple ranking strategies.

---

2. UML Class Diagram

https://lucid.app/lucidchart/480aba98-144f-4569-9ce5-4bab2252bc54/edit?invitationId=inv_ea559934-612b-4da1-b728-7bd6cf8539bd

or



---

3. Domain Model: 'JobSeekerProfile'

Class: JobSeekerProfile

Represents a user profile that can be ranked.

Attributes

- `id: String`

- `userId: String`
- `major: String`
- `graduationYear: int`
- `bio: String`
- `location: String`
- `phoneNumber: String`
- `skills: List<String>`
- `interests: List<String>`
- `resumeUrl: String`
- `portfolioUrl: String`
- `linkedInUrl: String`
- `githubUrl: String`
- `isStudent: bool`
- `jobType: String`
- `willingToRelocate: bool`
- `createdAt: DateTime`
- `updatedAt: DateTime`

Methods

- `toJson(): Map<String,dynamic>`
- `fromJson(json: Map<String, dynamic>): JobSeekerProfile`
- `copyWith(): JobSeekerProfile`

Role in UML

- Acts as the input for all ranking strategies.
- Strategies depend on this class but this class does not depend on strategies.

---

4. RankingStrategy Interface

Interface: RankingStrategy

Method

- `rank(profile: JobseekerProfile) : double`

This defines a contract for all ranking behaviors.
Each ranking class focuses on one dimension of the profile and returns a score.

UML Relationship

- Concrete ranking classes implement this interface.

---

5. Composite Ranking

Class: CompositeRankingStrategy

Attributes

- `strategies: List<JobSeekerRankingStrategy>`

Methods

- `addStrategy(strategy: RankingStrategy)`
- `removeStrategy(strategy: RankingStrategy)`
- `rank(profile: JobseekerProfile) : double`

Purpose

- Combine multiple strategy scores into a single score.
- Controls how ranking rules work together.
- You can plug/unplug strategies depending on the type of job or use-case.

UML Pattern

- **Composite Pattern**: CompositeRankingStrategy contains many RankingStrategies.

---

6. Concrete Ranking Strategies

Below this section we can find all ranking strategies included in the UML, with the fields they use.

6.1 **StudentStatusRankingStrategy**

Attributes

- `studentWeight: double`
- `nonStudentWeight: double`

Uses JobSeekerProfile fields

- `isStudent`

Purpose

- Score based on whether the user is a student or not.

6.2 **SkillRankingStrategy**

Attributes

- `skillWeights: Map<String, double>`

Uses JobSeekerProfile fields

- `skills: List<String>`

Purpose

- Score users based on the skills listed in their profile.

6.3 **JobTypeRankingStrategy**

Attributes

- `jobTypeWeights: Map<String, double>`

Uses JobSeekerProfile fields

- `jobType`

Purpose

- Score based on the user's desired job type (internship, full-time, etc.).

6.4 **ActivityRankingStrategy**

Attributes

- `isStudentWeight: double`
- `jobTypeWeight: double`

Uses JobSeekerProfile fields

- `createdAt`
- `updatedAt`
- `isStudent`
- `jobType`

Purpose

- Evaluates user activity recency and completeness.

6.5 **GPARankingStrategy**

Attributes

- `graduationYearWeight: double`
- `majorWeight: double`

Uses JobSeekerProfile fields

- `graduationYear`
- `major`

Purpose

- A proxy scoring method for experience/GPA-related metrics.

6.6 **LocationRankingStrategy**

Attributes

- `preferredLocations: List<String>`
- `locationWeights: double`

Uses JobSeekerProfile fields

- `location`
- `willingToRelocate`

Purpose

- Score based on geographic fit.

6.7 **InterestRankingStrategy**

Attributes

- `interestWeights: Map<String, double>`

Uses JobSeekerProfile fields

- `interests: List<String>`

Purpose

- Score based on interest alignment.

---

7. UML Relationships

1. CompositeRankingStrategy → RankingStrategy

Composition

- UML: Solid line with filled diamond at `CompositeRankingStrategy` pointing to `RankingStrategy`.
- CompositeRankingStrategy owns multiple strategies.
- If the composite is deleted, contained strategies are removed.

2. Concrete Strategies → RankingStrategy

Interface Realization

- UML: Dashed line with hollow triangle pointing to `RankingStrategy`.
- Each concrete strategy implements the interface.

3. CompositeRankingStrategy → Concrete Strategies

Association

- UML: Solid line from composite to each concrete strategy.
- Composite aggregates the concrete strategies for execution.

4. JobSeekerProfile

- Pure domain object; no structural dependencies to ranking classes.

---

8. Example Ranking Flow

1. A JobSeekerProfile is loaded.
2. CompositeRankingStrategy has several child strategies:
    - SkillRankingStrategy
    - LocationRankingStrategy
    - StudentStatusRankingStrategy
    - InterestRankingStrategy
    - etc.
3. CompositeRankingStrategy calls each:
    - `strategy.rank(profile)`
4. The scores are combined into one final ranking score.
5. Result is used for:
    - job recommendations
    - search ordering
    - profile matching

---

9. Conclusion

This UML and class structure provides:

- A modular ranking system
- A scalable and configurable approach to scoring
- Separation between profile data and ranking rules
- Flexibility for future extensions